COMPARISON OF RLL, STATE DIAGRAM, GRAFCET AND PETRI NET FOR THE REALIZATION OF LOGIC CONTROLLER

ZULHAIRI BIN OTHMAN

This thesis is submitted in partial fulfillment of the requirements for the award of the Degree of Master of Engineering (Electrical)

Fakulti Kejuruteraan
Kolej Universiti Teknologi Tun Hussein Onn

APRIL, 2005

ACKNOWLEDGEMENT

The writer is especially indebted to the supervisor of this thesis, Prof. Madya Dr. Zainal Alam bin Haron for his encouragement and guidance throughout the preparation of this thesis.

ABSTRACT

The strengths and weaknesses of popular plc programming tools may be a common knowledge to the experienced but that contention alone lacks depth to the many others. Several studies have presented weighted comparisons but focused on only two approaches at a time. The first part of this paper presents qualitative comparisons among the 4 most popular approaches: relay ladder logic (RLL), state diagram, grafcet and ordinary Petri net. Each approach is weighted by their understandability, efficiency and flexibility. It is the intent of the second part of this study to formulate a mix and match LLD realization method based on the compared model strengths and weaknesses. The proposed model is then compared with the internationally accepted Grafcet approach in light of the same criteria as the first part. An analysis entails on what has been gained and lost in the proposed approach. From these comparisons ultimately, it is hoped that the plc programmer is aware of the strengths and limitations of whichever programming approach chosen.

TABLE OF CONTENT

LIST OF TABLES

LIST OF FIGURES

CHAPTER I

PART 1: INTRODUCTION

Today's fast pace technology has generated a demand not only for the fastest and most efficient manufacturing system, but also for one that is highly flexible. While the general approach in the local scene tends towards upgrading of the system's hardware and software, it is often the chosen process and the method of process implementation themselves that dictate the long term success of any upgrades.

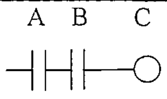
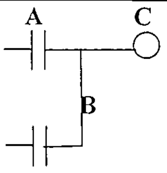
In the domain of programmable logic controller (plc), hardware and software enhancements are rolled out almost yearly, and in pace with the technological development of the larger sphere of digital world. But as in the case of the digital world, any upgraded system is only more efficient and effective if the user has the know-how and realizes the knowledge with methods that maximize the upgrade potential.

1.1 Evolution of LLD Implementation Methods

Ladder logic diagram (LLD) is the adopted 'software' for plc programming. It replaces the traditional pneumonic machine languages with electrical symbols common in hard-wired electrical circuit such as switches, relays, actuators etc.

The use of LLD as tool dates back in the 1950's – well before the introduction of a prototype plc. The obvious reason for this is LLD facilitates the actual realization of a designed system such as wiring and component arrangement. But more importantly, LLD possesses one-to-one relation to basic digital logic as in table 1.1.

Table 1.1: Direct logic representation of RLL

No	Logic Instruction	RLL
1	If A **AND** B, then C	
2	**If A** OR **B, then C**	

With the associations, the rules and methods of designing a discrete event system could adopt that of the digital counterpart. This important property of LLD is also the key to development and improvement of various methods of transforming discrete events and sequences to LLD models.

As the need for factory automation grew in the 1970's, so did the size and complexity of industrial automata and process requirements. One of the popular tools was the graphical state diagram and the tabulated state transition diagram. The two often works in tandem and will be referred to as simply state diagram method henceforth. This approach shifts the implementation focus from direct output implementation to process state and transition modeling.

Almost parallel, a French committee of academicians and industrialists presented their tool for LLD realization in 1977. This French standard (NFC-03-190), known as grafcet, was further enhanced in 1983. Grafcet's popularity eventually made it a basis for an international standard. The modeling standard went through a series of revisions until it is known as sequential function chart (SFC) in IEC 1131-3 standard – published in 1993. Since the basic model for LLD execution is similar in both grafcet and SFC, the approach will be referred to as grafcet. If the differences between the two models are distinguishable, it would be noted in the context.

Grafcet evolves from petri net model with several differences in interpretations and rules. Although grafcet is a subset of petri net model, it is only within the last decade that petri net itself is proposed as the logic controller implementation tool. Since there is no formal LLD transformation method from a petri net model that has been agreed, this study assumes similar LLD transformation as in grafcet and state diagram. The petri net model considered in this study is ordinary petri net. Several other forms of petri net model have been proposed for logic controller such as colored petri net, but the additional properties of these models – distributive and temporal for example – are not explicitly quantified in this study.

Besides the 4 formal methods, there are other approaches but most are either proprietary or unpopular in the manufacturing circle. The 4 selected methods are supported by the majority of plc manufacturers. This in turn ensures interoperability – a strong case for flexibility and adaptability in manufacturing system.

1.2 Objective

The brief historical review has shown the timeline of the dominant tools that corresponds to their era of industrial system size and complexity. However, the historical relation does not signify any weighted advantages or disadvantages since several approaches evolve from another out of a specific necessity.

This study aims to furnish a weighted comparison of the 4 selected tools in understandability, response time and adaptability. Several other comparison criteria will be discussed not as quantified arguments, but perceptive analysis.

With the exception of RLL approach, the other 3 models shares a common state/transition transformation into a working LLD in this study. There exist a direct and common proportionality then between the number of states (or places as called in petri net) and the number of elements in LLD for the 3 models. Consequently, the comparisons of understandability and flexibility of all approaches emphasize on graphical representation and not the resulting LLD. It is only when comparing time response, the number of nodes in LLD becomes a factor.

1.3 Brief Description of the Compared Methods

Relay Ladder Logic (RLL)

RLL in this paper refers to the method of LLD realization that transforms event sequence directly into LLD form without any tabulated or graphical modeling.

One such method is akin to a sequential try-and-error type LLD realization. This try-and-error design approach is the product of trying to relate process automata – in which actual sequence of activation and deactivation are important – with digital / binary logic – in which variable order of sequence does not change the output values. Although this method has undergone many improvements, it still lacks a systematic set of designed steps that could be applied to all process sequence of various types and complexity. Despite its heuristic approach, this method is often used as an introductory point to LLD programming due to its intuitively logical approach.

State Diagram

In a state diagram method, the discrete event sequence is transformed to a tabulated form of state tables. In some approaches, the table is then drawn in the form of graphical sequence.

One classic tabulated modeling is Huffman method. Unlike the sequential try-and-error approach, the designed steps in Huffman method are well defined and applicable to a broader range of process types. This approach starts by determining stable states of the discrete event sequence for all possible input combinations (sensory inputs for example) and then tabulates them. The stable states are then combined according to a set of merging rules. Next, the merged states are assigned with flip flop outputs in accordance with the sequence. The final output functions are drawn out of the Karnough map of the flip flop outputs.

Note that the key to Huffman method is its successful way of correlating a sequential output with the established mathematics of combinational or Boolean logics. A detailed tutorial of Huffman method is presented in [12].

In the state diagram graphical form or henceforth referred to as state diagram, each tabulated stable states is drawn as a circle labeled with its outputs' names. The sequential relation of any 2 state circles is represented by directed arcs (arrows) labeled with the input that triggers the state. LLD is realized from the graphical model by assigning a flip flop output for each state. The flip flop output is then activated by a combinational function of its inputs and the link upstream states. In plc programming, this LLD implementation from the state diagram is similar to the method used by grafcet.

Grafcet and Petri net

Grafcet is essentially a subset of Petri net modeling but differs in interpretation and its LLD implementation rules. Both graphical forms bears close resemblance to the graphics of state diagram with several differences as discuss below.

A grafcet step, represented by a square as in figure 1.1, models only the desired output(s) that is/are active according to the process sequence. In discrete event system, one sequential step corresponds to unique state of the sequence. Therefore, a grafcet step is often interpreted similar to a state diagram's state.

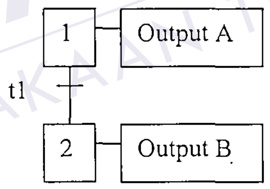


Figure 1.1: Simple grafcet graphical form

The difference between the two models' 'state' is best explained with a concurrent system. 2 events are concurrent if the event is causally independent. Consider a process sequence that starts out in a single path as in figure 1.1 but then branches out to 2 concurrent paths as soon as actuator (B) completes extending and the corresponding sensor (b0) turns on, as in figure 1.2.

The grafcet model and a state diagram model of figure 1.2 prior to the activation of b0 looks structurally similar. But after b0 turns on, the grafcet model takes on only 2 paths with each path indicating only the active outputs belonging to that particular branch.

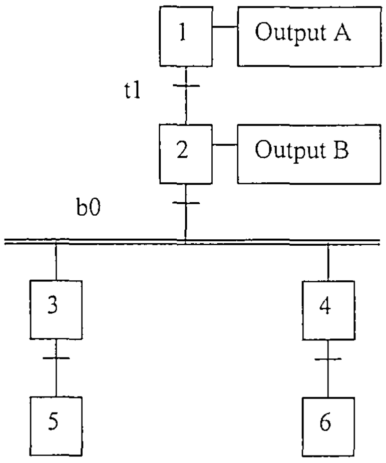


Figure 1.2: Grafcet model of branch out sequence

In a state diagram representation however, the structure after b0 would show a web of interlinked states, devoid of any distinctive sequential paths. Each state in the state diagram shows all the possible and stable combinations of active outputs. Consequently, outputs from the 2 branched out paths are intermixed in labels on each state circles. This difference is further highlighted in the discussion part of part 1.

In Petri net, the counterpart of a grafcet step is called a place – represented by a circle as in figure 1.3. Unlike grafcet's singular purpose of logical sequence implementation, a Petri net model could also be applied to a broader range of system design and analysis such as in data communication and manufacturing resource planning. In the narrowed scope of LLD implementation, a Petri net model could be interpreted similar to grafcet.

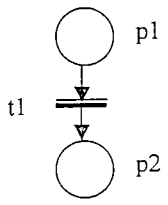


Figure 1.3: Simple Petri net graphical form

One important difference is that one Petri net place corresponds to the status of only one resource or output. Structurally, a grafcet sequence could have any outputs repeatedly active at another step(s) as called for by the event sequence. But in a Petri net representation of the same sequence, only one place bearing the label of that output will be shown. The place would have more that one directed arc both going into and coming out of the place. Other differences are discussed in the discussion section of this study.

The grafcet and SFC method are elaborated in [9] whereas a tutorial paper of grafcet is presented in [6]. [10] is tutorial paper on ordinary Petri net.

CHAPTER II

LITERATURE REVIEW

By implementing ladder logic diagram (LLD) based on process automata, the relay ladder logic (RLL) achieves efficiency in hard wiring and circuit element usage. RLL was the natural adopted approach of plc programming in early 1970's since it was hard-wired logic that plc was intended to replace. However, as more process sequences were automated and became more complex, implementing LLD via the RLL method became heuristic.

The notion of state in graphical models such state diagram proved to be the key when dealing with large complex process sequence. The realization focus shifted from modeling around process automata to the discrete states of a process. It is the process state's that in turn, activates or deactivates process automata. This concept permeated to all other LLD realization approaches. In the state diagram method, the concept of state was influenced by works from C.F. Moore in "Gedanken – Experiments on Sequential Machines" in "Automated Studies" – a Princeton University Press (N.J.) publication.

Inasmuch as state diagram approach proved successful in unraveling the complex sequences of discrete event system, the approach itself could very easily turn out to be a complex web of states and transitions. The reason for this is its inclusion of all possible internal states of the system to its graphical representation. Consequently, concurrent process or sequence could not be shown graphically as separate but parallel sequential paths.

This weakness, among others, has led to C.A. Petri's proposal of petri net concept in his 1962 PhD dissertation: "Kommunikation mit Automaten" – submitted to the University of Damstadt, West Germany. Petri net as a LLD tool was almost obscure among plc programmers in that first plc decade. The model later formed the basis of French formal plc tool called grafcet. While typical application of petri net is total system modeling, grafcet as a model specifically aims to facilitate LLD implementation. The analytical and testable strength of petri net was originally never adapted to the grafcet model.

It was only within the last decade that the interest in petri net as logic controller tool gained momentum. Publications of petri net as logic controller tool could be found in [1] and [15].

There exist other tools but not as popular for LLD implementations such as finite state diagram (FSM) and function block diagram (FBD). Only the 4 approaches would be considered in this study owing to their dominance.

A qualitative study that includes the 4 approaches is useful as a guide for LLD design. A quantified comparison between RLL and petri net by K. Venkatesh and M.C. Zhou focuses on design complexity, time response and adaptability [14]. J.S. Lee and P.L. Hsu devise an if-then transformation of RLL and petri net for quantitative comparisons in [8]. The focus is mainly on the time response of the 2 models.

One comparison that includes all 4 approaches is authored by K. Feldman and A.W. Colombo in [7]. Since it's not the aim of that paper to compare the four implementation tools, it does not present a qualitative analysis. It is the intent of this study to fulfill that need.

CHAPTER III

METHOD

The objective of this study is to present weighted analysis of each approach's understandability, response time and flexibility. The two criteria of understandability and flexibility pertain to the human interface aspect of the model whereas response time relates to hardware implementation of the approach. In that light, only response time is evaluated quantitatively whereas the understandability and flexibility are evaluated perceptively from the results.

A test case similar to the example presented in [4] and [6] is used as in figure 3.1 below. This process offers simple and concurrent process that is independent and parallel to each other.

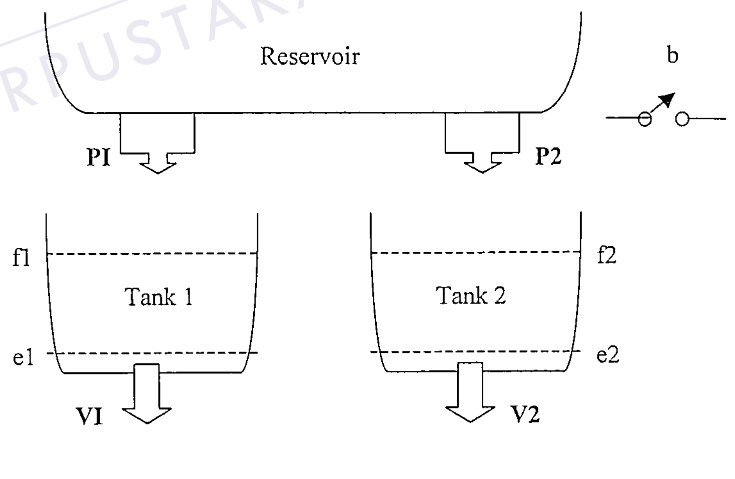


Figure 3.1: Concurrent tank filling process

The concurrent sequences of the test case of figure 3.1 start by pressing push button 'b'. When the empty sensors, e1 and e2, detect no water (→ e1' and e2') the pumps, P1 and P2, are activated to fill up tank 1 and tank 2 in parallel but independently. The pumps turn off when the full sensors, f1 and f2, detect water. This in turn triggers the activation of the valves, V1 and V2, independently to drain out water from the tanks. The cycle for each tank continues when the respective empty sensor detects the tank is empty. Note the notation used in this paper: e1' indicates a 'Not' e1.

3.1 Understandability

Although this criterion is merely perceptive, the graphical representation of the approaches should conform to the accepted norms of graphical models such as flowchart method. One of the most important properties of these accepted models is the ability to present sequential flow in hierarchal fashion. Another property that promotes understanding is the ability to present concurrent and independent sequence unambiguously.

3.1.1 Hierarchy

The measure of hierarchy for each approach graphical representation is a top to bottom flow of sequence. The graphical models should show the 'when', 'where', and 'how' the flow relate to the actual manufacturing rather than merely on 'how' for example. Where flow diverges, converges or becomes too large to fit, clear markings should be used to indicate the relation of the detached sequences. These properties are essential for fast and accurate human interfacing – for example maintenance, modification or monitoring/testing purposes.

3.1.2 Concurrency

As in the case of the test case, the 2 independent and concurrent sequences should also be presented graphically as independent and parallel; the graphical flow should not be ambiguous where the sequence diverges and converges for the test case.

For the test case of figure 3.1, the measure of concurrency is a graphically distinctive path for a parallel and independent process and a sharing of common nodes when the two paths should converge.

3.2 Response Time

A plc execution cycle reads and updates each element in LLD sequentially from top to bottom. The response time of execution could then be measured by the number of elements or nodes in the rungs of LLD program.

In RLL, a combinational function of a process automa could be directly implemented in LLD by the correlation of table 1.1. Typically, an RLL output has the following form:

$\text{Output} = (t_{on} + \text{Output}_{interlock}) * t_{off}'$

These direct forms are not shown for state diagram, grafcet and petri net because of these approaches to modeling process automata are in the form of process state and transitions instead of focusing on process output. The bipartite nature of the 3 models graphical forms may mislead the true meaning of the logic representation if it is tabulated in the same table 3.1.

For state diagram, grafcet and petri net, the realization of LLD from the graphical states and transitions are considered similar in this study. In these approaches, the firing of one transition activates the next immediate state(s) and deactivates the previous. Note that this generalized statement does not accurately reflect the token game of petri net model but its final LLD outcome is similar.

Borrowing petri net's mathematical relation for a firing sequence, the equation is as follow:

$m(p) = m_{previous}(p) + O(t,p) - I(p,t)$

where $m(p)$ is the new marking
$m_{previous}(p)$ is the previous marking
$O(t,p)$ is the postset of transition t
$I(t,p)$ is the preset of transition t

During the LLD transformation of this firing sequence, the equation could be translated as follows. The firing latches on the next immediate state O(t,p) and unlatches previous state I(t,p). The LLD program is now pointing to a new marking m(p), which is the sequential upstream marking from the previous marking $m_{previous}(p)$.

For a transition that activates (latches) a single state and deactivates (unlatches) one state – as in the case for one state machine (SM), the LLD transformation consists of 3 elements as in figure 3.2. In this study, a plc 'Keep' or RS flip flop is used for the latching and unlatching of the states.

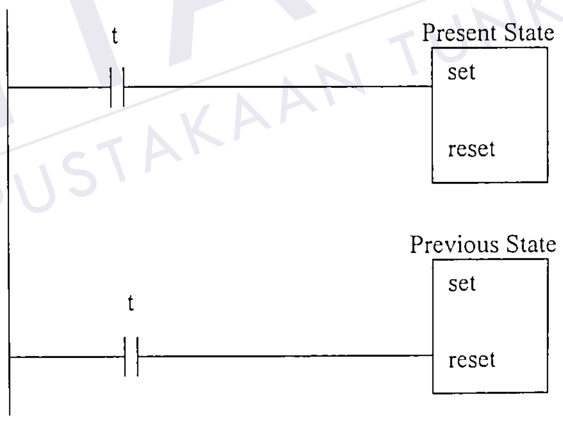


Figure 3.2: Activation and deactivation of one state machine

The transition t is only firable when the immediate preceding states are active and when its receptivity is true.

$t = m_{previous}(p)$. receptivity

For the SM example, the LLD transformation has 3 elements, assuming a single receptivity for the transition. This is shown in figure 3.3.

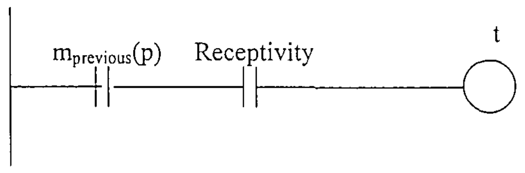


Figure 3.3: Enabling a firing in LLD

For the SM example, a single firing is transformed into LLD with 6 elements. This number is consistent for state diagram, grafcet and petri net approach.

This transformation method is applied for the 3 approaches. The total element of each LLD is then counted and represents a figure for each of the models.

For the sake of clarity, the LLD initialization rungs are not shown in any of the results. The basic ladder diagram structure for all initialization sequences is assumed to produce similar quantity of components used. This is justified because the objective of initialization is to ensure all state controller contacts initialize and that all actuators have reverted to their rest positions. In the case of RLL where the state concept is not used, the initialization sequence ensures that the master controller contacts are initialized.

3.3 Flexibility

The flexibility of a tool determines its adaptive and responsive capability for human interfacing. For the test case, an additional test is designed to compare the flexibility of the models. A process change to the tank filling process requires the tank f1 discharge via V1 to stop immediately on the issuance of an external event 'c' from a larger process . Tank 1 should then be filled up again even though it is not empty. Flexibility is then measured from the resulting graphical models of each approach.

3.3.1 Efficiency of Process Change

The number of changes to each approach's graphical representation is proportional to the changes required in LLD. A quantitative comparison of LLD changes presents a measure of efficiency in carrying out a process change.

CHAPTER IV

RESULTS

4.1 RLL

4.1.1 Test case graphical model

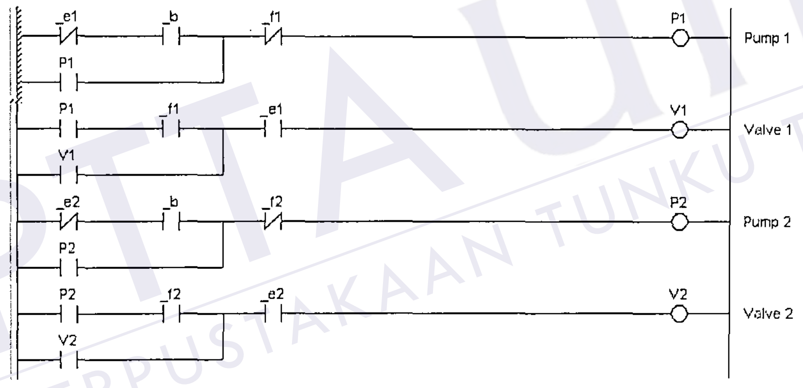


Figure 4.1: RLL graphical representation

4.1.2 Response Time

The LLD of figure 4.1 consists of 20 nodes.

4.1.3 Flexibility: Efficiency

A normally close contact 'c' is 'And' together to 'e1' in the rung for Valve 1 of figure 4.1. The equation for Valve 1 is as follows:

REFERENCES

[1] Adamski, M.A. and Monteiro,J.L., "Rule-based Formal Specification and Impelementation of Logic Controllers Programs," Proceedings of IEEE Int.Symp. on Industrial Electronics, vol. 2, pp700-705, July 1993.

[2] Andreu, D. and Pascal, J.C., "Fuzzy Petri Net-based Programmable Logic Controller," IEEE Transaction on Syst., Man, and Cybernetics-Part B: Cybernetics, vol. 27, no. 6, Dec 1997.

[3] Azevedo, J.L. and deOliveira, J.P., "The Grafcet's Macro-Action Concept: An Implementation View," Proceedings of the 7th IEEE Int. Conf. on Emerging Technologies and Factory Automation, vol.2, pp1275-1279, EiFA 1999.

[4] Charbonnier, F., Alla, H. and David, R., "The Supervised Control of Discrete-Event Dynamic Systems," IEEE Transactions on Cont. Syst. Technology, vol. 7, no. 2, pp175-187, March 1999.

[5] Chen, S.M., "Weighted Fuzzy Reasoning using Weighted Fuzzy Petri Nets," IEEE Transaction on Knowledge and Data Engineering, vol. 14, no. 2, pp386-397, March/April 2002.

[6] David, R., "Grafcet: A Powerful Tool for Specification of Logic Controllers," IEEE Transaction on Cont. Syst. Technology, vol. 3, no. 3, pp253-268, Sept 1995.

[7] Feldmann, K., Colombo, A.W., Schnur, C. and Stockel, T., "Specification, Design, and Implementation of Logic Controllers based on Colored Petri Net Models and the Standard IEC 1131 Part I: Specification and Design," IEEE Transaction on Cont. Syst. Technology, vol. 7, no. 6, pp657-665, Nov 1999.

[8] Lee, J.S. and Hsu, P.L., "A New Approach to Evaluate Ladder Logic Diagrams and Petri Nets via the If-Then Transformation," Proceedings of IEEE Int. Conf. on Man and Cybernetics, vol. 4, pp2711-2716, Oct 2001.

[9] Lewis, R.W., "Programming Industrial Control Systems using IEC1131-3: Revised Edition," The Institution of Electrical Engineers, 1998.

[10] Murata, T., "Petri Nets: Properties, Analysis and Applications," Proceedings of the IEEE, vol. 77, no. 4, pp541-578, April 1989.

[11] Parr, E.A., "Programmable Controllers: An Engineer's Guide," 2nd Edition, Newness, 1999.

[12] Pessen, David W., "Industrial Automation: Circuit Design and Components," John Wiley and Sons, p168-218, 1989.

[13] Seshu, S., Miller, R.E. and Metze, G., "Transition Matrices of Sequential Machines," IRE Transaction on Circuit Theory, pp5-12, March 1959.

[14] Venkatesh, K., Zhou, M.C. and Caudill, R.J., "Comparing Ladder Logic Diagram and Petri Nets for Sequence Controller Design through a Discrete Manufacturing System," IEEE Transaction on Industrial Electronics, vol. 41, no. 6, Dec 1994.

[15] Wegrzyn, A. and Wegrzyn, M., "Petri Net-based Specification, Analysis and Synthesis of Logic Controllers," Proceedings of the 2000 IEEE Int. Symp. on Industrial Electronics, vol. 1, pp20-26, ISIE 2001.